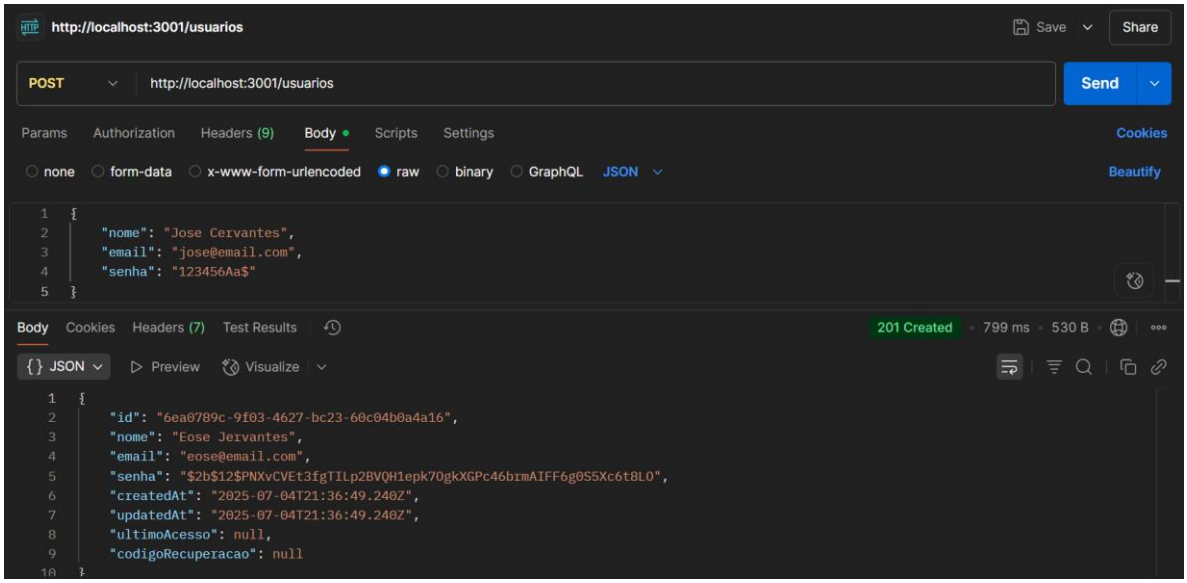
Centro Universitário UniSenac – Campus Pelotas

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

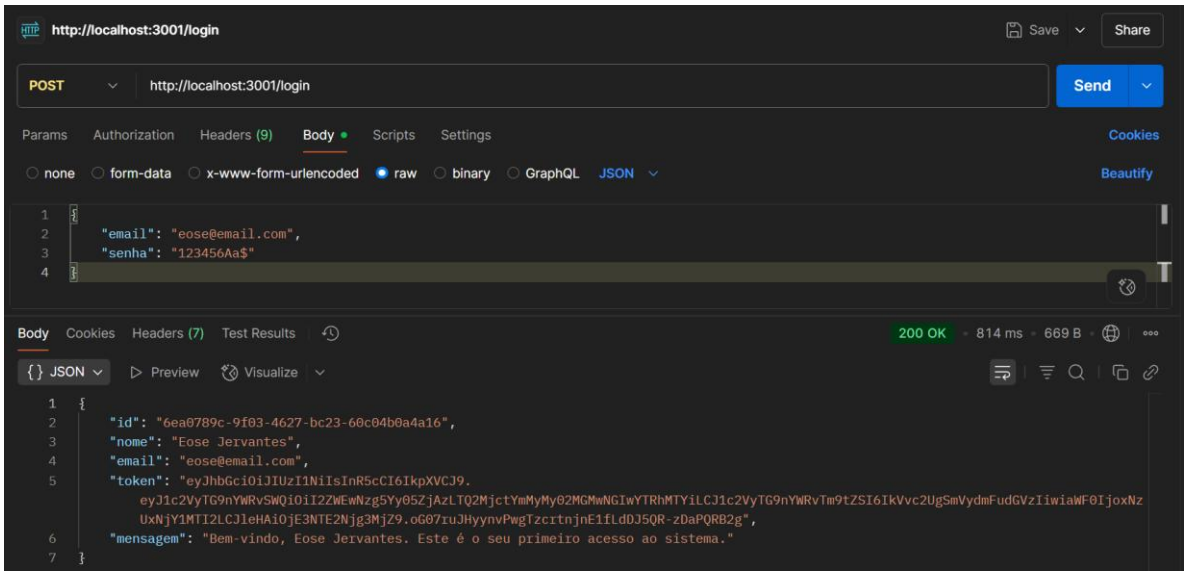
Disciplina: Desenvolvimento de Serviços e APIs – Prof. Edécio

Trabalho #3: Recursos de Segurança de Sistemas no Desenvolvimento de APIs

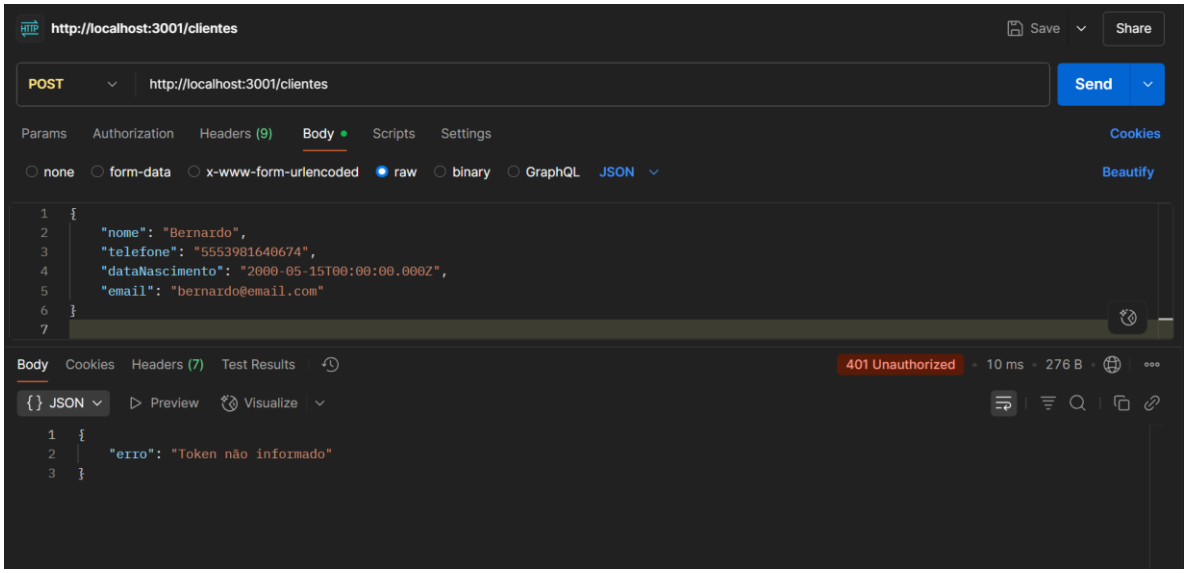
1. Tela de Cadastro de Usuários com Criptografia da Senha (Insomnia)



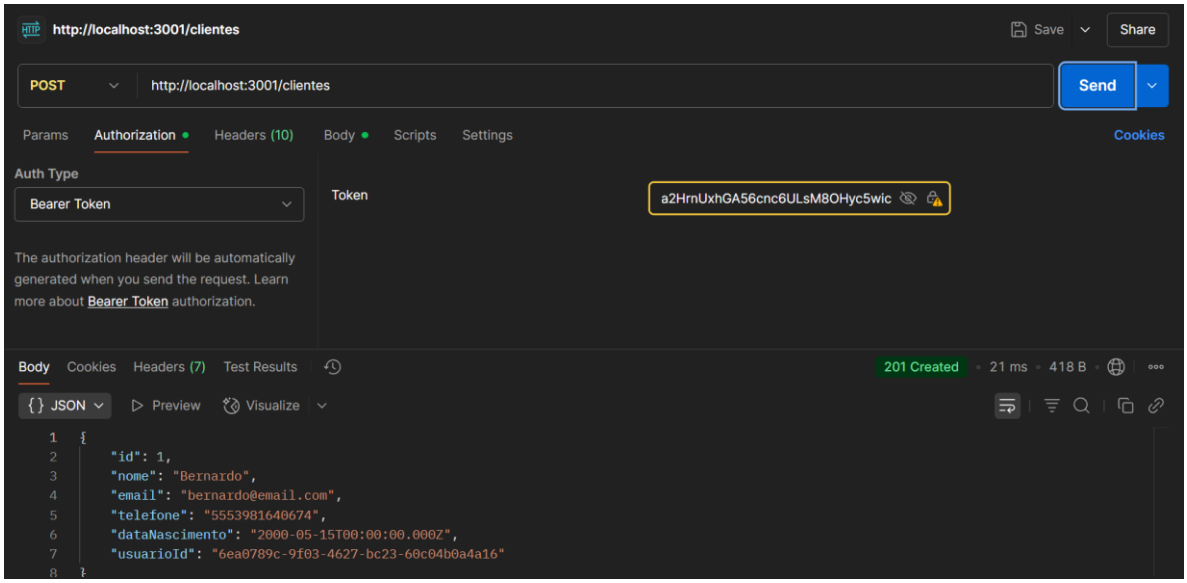
2. Tela de Login do Usuario com Geração do Token de Acesso



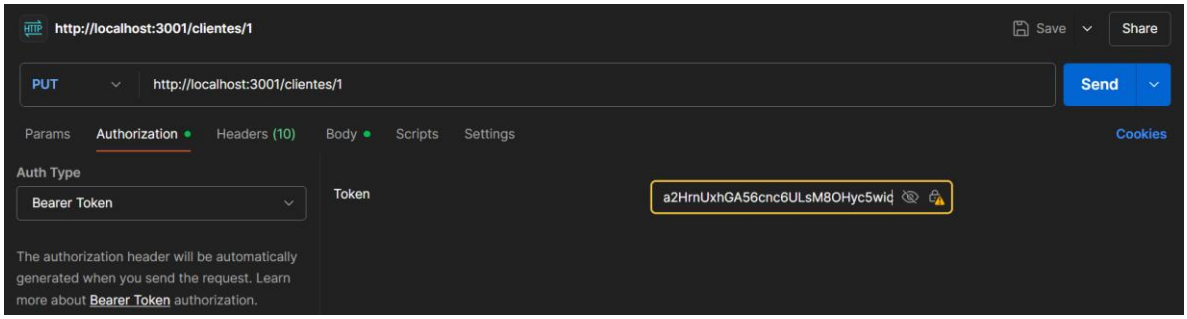
3. Exemplo de Rota com Exigência do Envio do Token

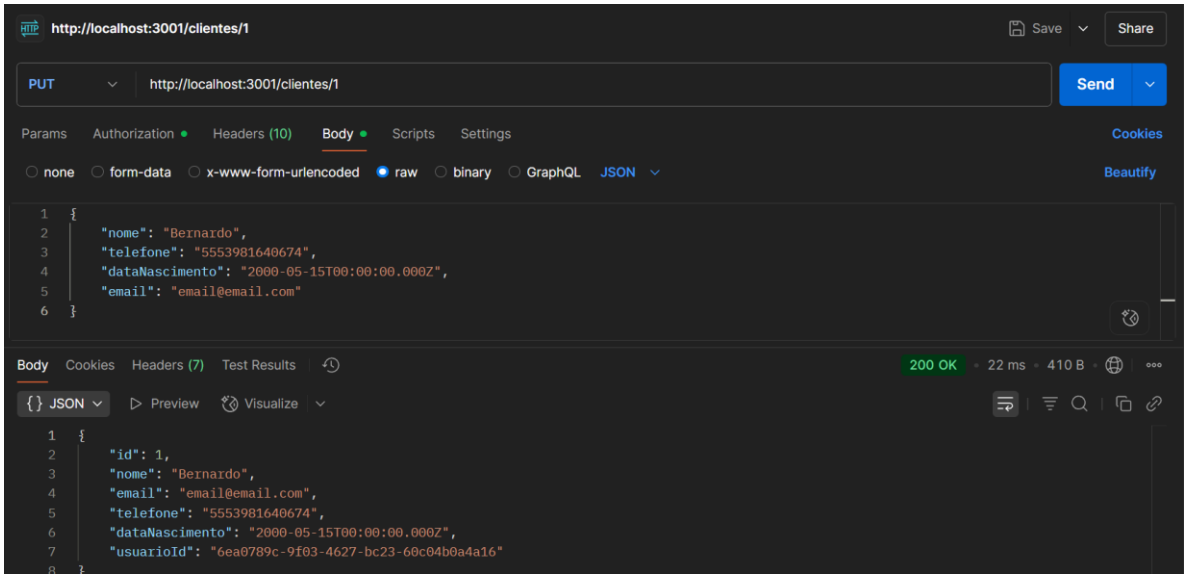


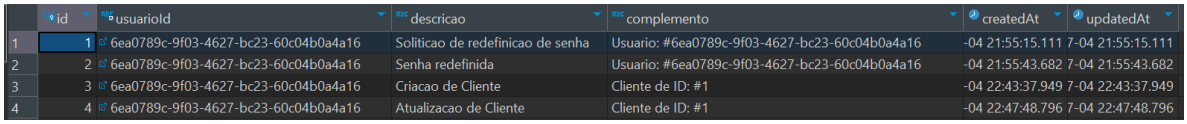
4. Acesso a mesma rota com o envio do Token



5. Acessar rota que gera log no sistema (fazer select dos logs)







	id	usuarioId	descricao	complemento	createdAt	updatedAt
1	1	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Soliticao de redefinicao de senha	Usuario: #6ea0789c-9f03-4627-bc23-60c04b0a4a16	-04 21:55:15.111	7-04 21:55:15.111
2	2	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Senha redefinida	Usuario: #6ea0789c-9f03-4627-bc23-60c04b0a4a16	-04 21:55:43.682	7-04 21:55:43.682
3	3	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Criacao de Cliente	Cliente de ID: #1	-04 22:43:37.949	7-04 22:43:37.949
4	4	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Atualizacao de Cliente	Cliente de ID: #1	-04 22:47:48.796	7-04 22:47:48.796

6. Código da rotina de backup dos dados do sistema

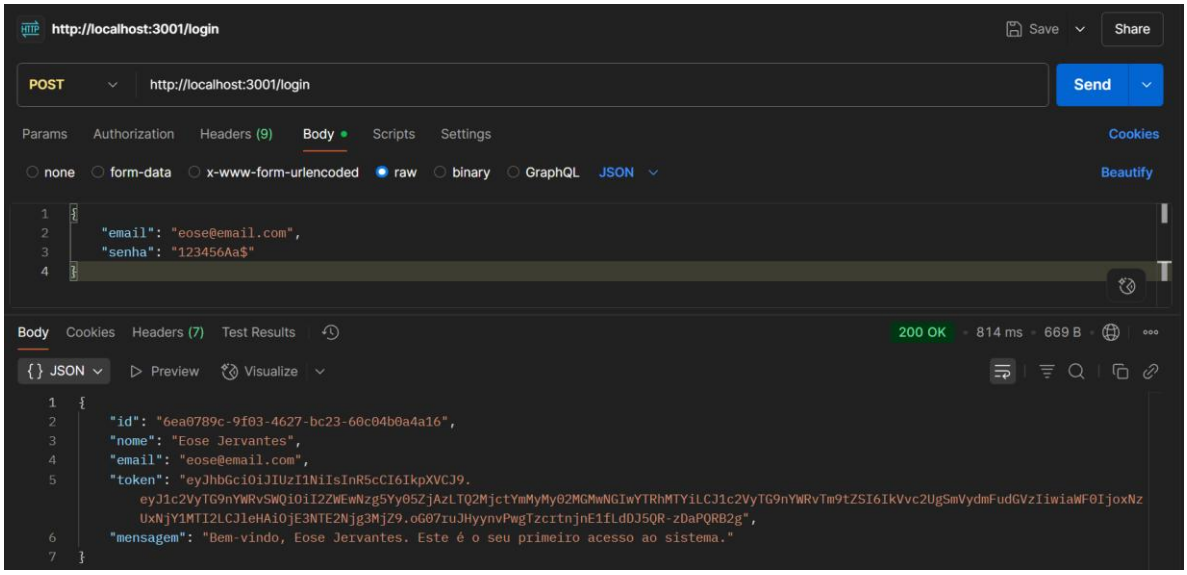
```
router.get('/backup', async (req, res) => {
 try {
  const usuarios = await prisma.usuario.findMany()
  const clientes = await prisma.cliente.findMany()
  const filmes = await prisma.filme.findMany()
  const locacoes = await prisma.locacao.findMany()
  const logs = await prisma.log.findMany()

  const dadosBackup = {
   usuarios,
   clientes,
```

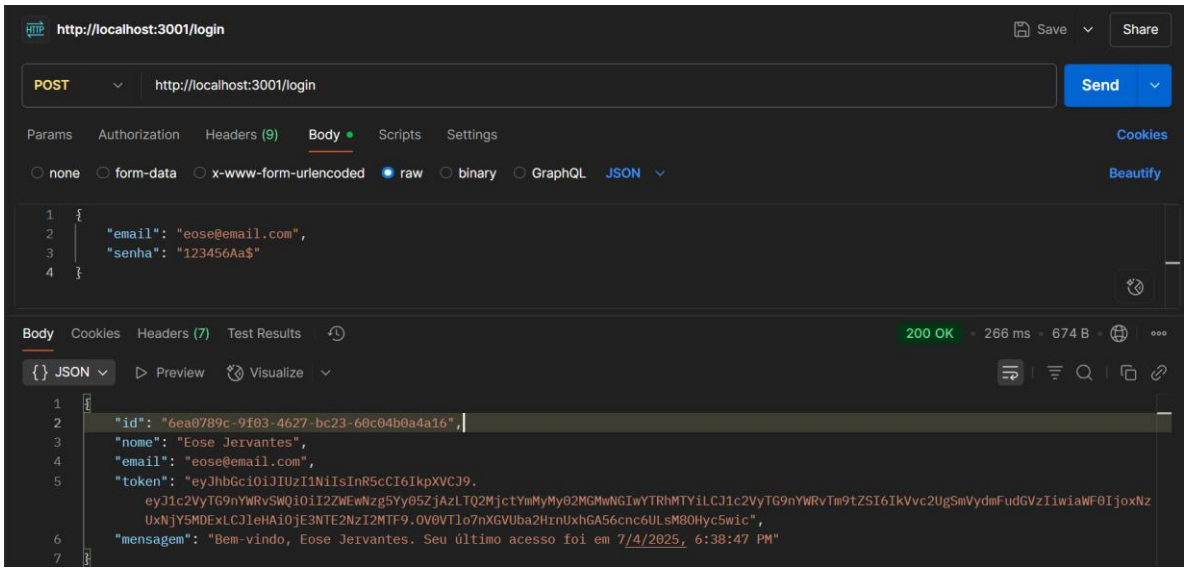
```
    filmes,
    locacoes,
    logs,
    dataBackup: new Date().toISOString()
   }

   const caminho = path.resolve(__dirname, '../../backup.json')
   fs.writeFileSync(caminho, JSON.stringify(dadosBackup, null, 2))

   res.json({ mensagem: 'Backup gerado com sucesso!', arquivo: 'backup.json'
})
  } catch (erro) {
   console.error(erro)
   res.status(500).json({ erro: 'Erro ao gerar o backup.' })
  }
})

7. Código do Recurso Extra de Segurança (#1)
router.post("/", async (req, res) => {
  const { email, senha } = req.body

  const mensagemPadrao = "Login ou senha incorretos"

  if (!email || !senha) {
   res.status(400).json({ erro: mensagemPadrao })
   return
  }

  try {
   const usuario = await prisma.usuario.findFirst({
    where: { email }
   })

   if (usuario == null) {
    res.status(400).json({ erro: mensagemPadrao })
    return
   }
```

```
  if (bcrypt.compareSync(senha, usuario.senha)) {
    const token = jwt.sign({
      userLogadoId: usuario.id,
      userLogadoNome: usuario.nome
    },
      process.env.JWT_KEY as string,
      { expiresIn: "1h" }
    )

    const ultimoAcessoAnterior = usuario.ultimoAcesso

    await prisma.usuario.update({
      where: { id: usuario.id },
      data: { ultimoAcesso: new Date() }
    })

  const mensagemBoasVindas = ultimoAcessoAnterior
    ? `Bem-vindo, ${usuario.nome}. Seu último acesso foi em ${new
Date(ultimoAcessoAnterior).toLocaleString()}`
    : `Bem-vindo, ${usuario.nome}. Este é o seu primeiro acesso ao sistema.`

    res.status(200).json({
      id: usuario.id,
      nome: usuario.nome,
      email: usuario.email,
      token,
      mensagem: mensagemBoasVindas
    })
  } else {
    const descricao = "Tentativa de acesso ao sistema"
    const complemento = "Usuário: " + usuario.id + " - " + usuario.nome

    const log = await prisma.log.create({
      data: { descricao, complemento, usuarioId: usuario.id }
    })

    res.status(400).json({ erro: mensagemPadrao })
  }
```

```
  } catch (error) {
    res.status(400).json(error)
  }
})
```

8. Tela/demonstração do Recurso Extra de Segurança (#1)





9. Código do Recurso Extra de Segurança (#2)

```
const redefinirSenhaSchema = z.object({
  email: z.string().email({ message: "E-mail inválido" }),
});
```

```
const redefinirSenhaValidaSchema = z.object({
  email: z.string().email({ message: "E-mail inválido" }),
  senha: z.string(),
  codigo: z.string().length(6, { message: "Código deve ter 6 caracteres" }),
});

function gerarEmailRedefinicaoHTML(usuario: any, codigo: string) {
  const html = `
    <html>
      <body style="font-family: Helvetica, Arial, sans-serif; line-height: 1.6;">
        <h2>Locadora Avenida - Recuperação de Senha</h2>
        <p>Olá, <strong>${usuario.nome}</strong>.</p>
        <p>Recebemos uma solicitação para redefinir sua senha.</p>
        <p><strong>Este é o seu código de verificação:</strong></p>
        <div style="font-size: 24px; font-weight: bold; margin: 20px 0; color:
#2c3e50;">
          ${codigo}
        </div>
        <p>Este código é válido por 15 minutos.</p>
        <p>Se você não solicitou a redefinição de senha, ignore este e-mail.</p>
        <br>
        <p>Atenciosamente,</p>
        <p><strong>Equipe Locadora Avenida</strong></p>
      </body>
    </html>
  `;
  return html;
}

const transporter = nodemailer.createTransport({
  host: process.env.SMTP_HOST,
  port: Number(process.env.SMTP_PORT),
  secure: false,
  auth: {
    user: process.env.SMTP_USER,
    pass: process.env.SMTP_PASS,
  },
});
```

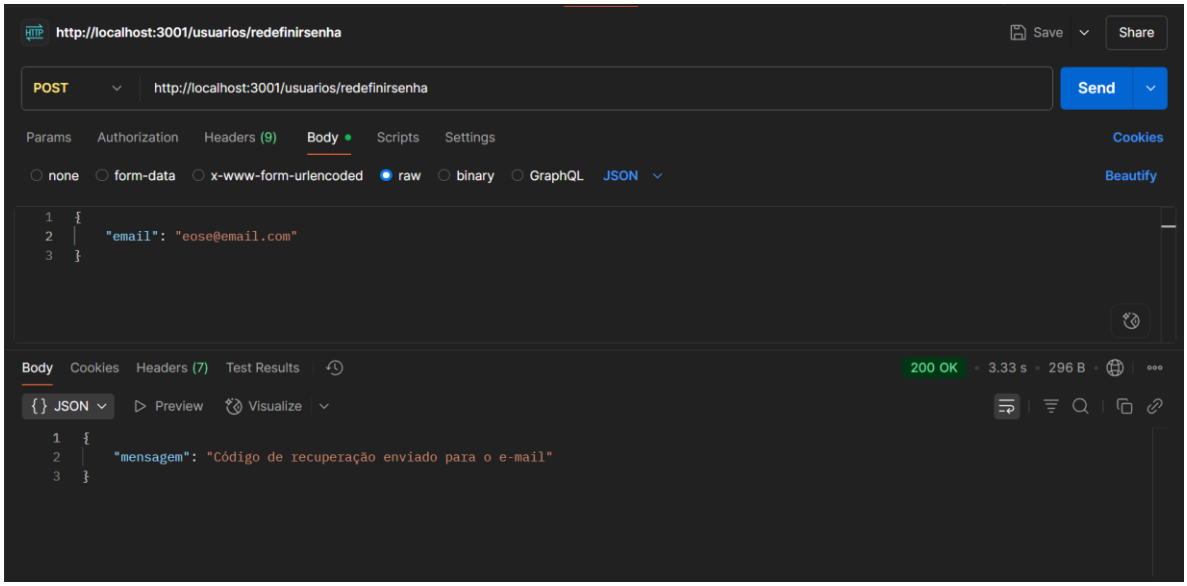
```
async function enviaEmailRedefinicao(usuario: any, codigo: string) {
 const mensagem = gerarEmailRedefinicaoHTML(usuario, codigo);

 const info = await transporter.sendMail({
  from: "Locadora <locadora-avenida@no-reply.com>",
  to: usuario.email,
  subject: "Recuperação de Senha",
  text: "Segue seu token para redefinicao de senha.", // Corpo simples
  html: mensagem, // Corpo HTML
 });

 console.log("E-mail enviado:", info.messageId);
}

router.post("/redefinirsenha", async (req: any, res: any) => {
 const valida = redefinirSenhaSchema.safeParse(req.body);

 if (!valida.success) {
  return res.status(400).json({ erro: valida.error });
 }

 const { email } = valida.data;

 try {
  const usuario = await prisma.usuario.findUnique({
   where: { email },
  });

  if (!usuario) {
   return res
    .status(400)
    .json({ erro: "Usuário não encontrado com este e-mail" });
  }

  const codigo = Math.floor(100000 + Math.random() * 900000).toString();

  await prisma.usuario.update({
```

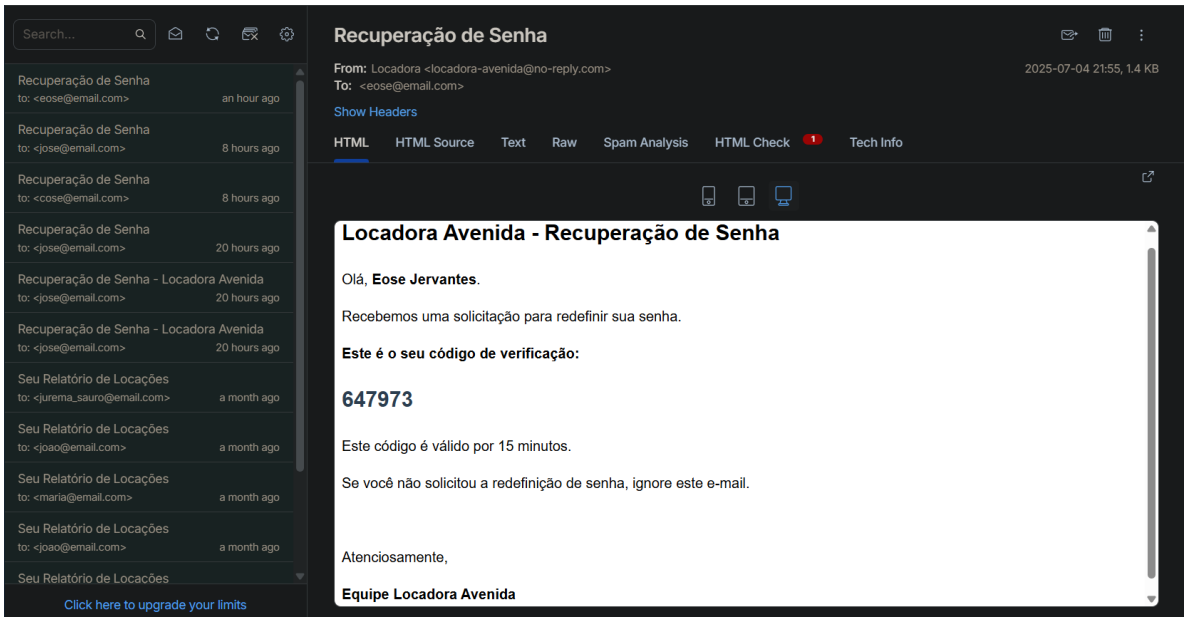
```
      where: { email },
      data: {
        codigoRecuperacao: codigo,
      },
    });

    const descricao = `Soliticao de redefinicao de senha`;
    const complemento = `Usuario: #${usuario.id}`;

    const log = await prisma.log.create({
      data: { descricao, complemento, usuarioId: usuario.id },
    });

    await enviaEmailRedefinicao(usuario, codigo);

    res.status(200).json({
      mensagem: "Código de recuperação enviado para o e-mail",
    });
  } catch (error) {
    res.status(500).json({ erro: "Erro ao processar solicitação" });
  }
});

router.post("/redefinirsenha/validar", async (req: any, res: any) => {
  const valida = redefinirSenhaValidaSchema.safeParse(req.body);

  if (!valida.success) {
    return res.status(400).json({ erro: valida.error });
  }

  const { email, senha, codigo } = valida.data;

  try {
    const usuario = await prisma.usuario.findUnique({
      where: { email },
    });

    if (!usuario) {
```

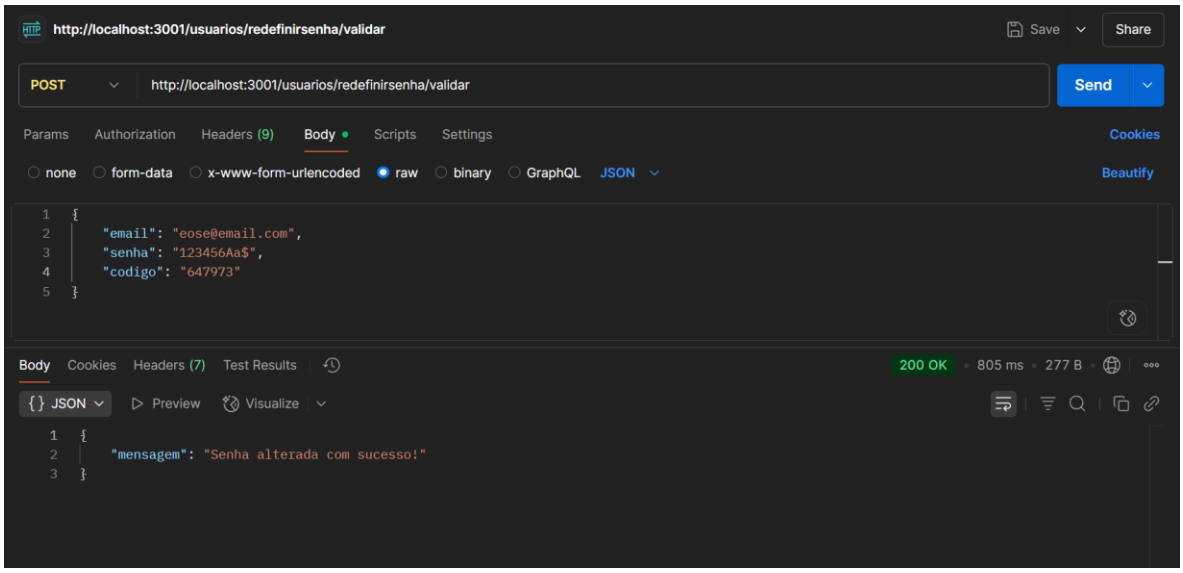
```
  return res
    .status(400)
    .json({ erro: "Usuário não encontrado com este e-mail" });
 }

 if (usuario.codigoRecuperacao !== codigo) {
  return res.status(400).json({ erro: "Código de recuperação inválido" });
 }

 const mensagensErro = validaSenha(senha);

 if (mensagensErro.length > 0) {
  res.status(400).json({ erro: mensagensErro.join("; ") });
  return;
 }

 const salt = bcrypt.genSaltSync(12);
 const hash = bcrypt.hashSync(senha, salt);

 await prisma.usuario.update({
  where: { email },
  data: {
   senha: hash,
   codigoRecuperacao: null,
  },
 });

 const descricao = `Senha redefinida`;
 const complemento = `Usuario: #${usuario.id}`;

 const log = await prisma.log.create({
  data: { descricao, complemento, usuarioId: usuario.id },
 });

 res.status(200).json({
  mensagem: "Senha alterada com sucesso!",
 });
} catch (error) {
```

```
  res.status(500).json({ erro: "Erro ao processar solicitação" });
 }
});
```

10. Tela/demonstração do Recurso Extra de Segurança (#2)







	id	usuarioId	descricao	complemento	createdAt	updatedAt
1	1	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Soliticao de redefinicao de senha	Usuario: #6ea0789c-9f03-4627-bc23-60c04b0a4a16	-04 21:55:15.111	7-04 21:55:15.111
2	2	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Senha redefinida	Usuario: #6ea0789c-9f03-4627-bc23-60c04b0a4a16	-04 21:55:43.682	7-04 21:55:43.682
3	3	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Criacao de Cliente	Cliente de ID: #1	-04 22:43:37.949	7-04 22:43:37.949
4	4	6ea0789c-9f03-4627-bc23-60c04b0a4a16	Atualizacao de Cliente	Cliente de ID: #1	-04 22:47:48.796	7-04 22:47:48.796